02142

Eur päisches Patentamt
Eur pean Patent Office
Office eur péen des brevets

Bescheinigung Certificate Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. Patent application No. Demande de brevet n°

02102738.8

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

Anmeldung Nr:
Application no.: 02102738.8
Demande no:

Anmeldetag:
Date of filing: 12.12.02
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

AGFA-GEVAERT
Septestraat 27
2640 Mortsel
BELGIQUE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

Method for sending messages in a computer network

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

H04L29/06

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SI SK

[DESCRIPTION]

FIELD OF THE INVENTION

The present invention relates to a system and a method for establishing communications between a server computer and a client computer via a network such as the Internet.

BACKGROUND OF THE INVENTION

In the printing and publishing environment, different players interact in order to obtain print and publishing products, such as magazines, catalogues, promotional, corporate, book or specialty products in offset, flexo, screen, digital, sheet- or web-fed printing. The main players that interact, in what is called in this document the "Graphic Enterprise", are the print buyer (or customer), the people in the workcenter, and the customer service representative who is the communicator between the first two main players. Different tools are used within the Graphic Enterprise, such as mail servers, pre-press workflow systems (such as Apogee Series 3 and Apogee X from Agfa), cost estimation modules, Management Information Systems (MIS), Job Definition Format (JDF) devices, etc. However, the communication between these tools and between the different players in the Graphic Enterprise is often unstructured.

There is thus a need for improved communication between the different players and the tools within the Graphic Enterprise.

SUMMARY OF THE INVENTION

The present invention is a method for sending a message from a server computer to a client computer via a network as claimed in independent claim 1. Preferably, a method in accordance with the invention is implemented by a set of computer programs as claimed in

claim 9. The invention also includes a method for sending a message by a server computer as claimed in independent claim 11, a method for receiving a message by a client computer as claimed in independent claim 14 and the corresponding computer programs. The invention further includes data processing systems (such as a computer network system, a server computer or a client computer) comprising means for carrying out these methods and computer readable media comprising program code adapted to perform these methods. Preferred embodiments of the invention are set out in the dependent claims.

In a method in accordance with the invention, it is preferred to use queues to transmit a message from one computer to another one. A queue is a sequence of messages or jobs held in auxiliary storage awaiting transmission or processing. In a preferred embodiment of the invention, the sending computer and the receiving computer are both connected to a computer network and they both have an input queue and an output queue. Messages are transmitted between these queues of the computers. Advantageously, these queues are physically associated with the corresponding computers. That a computer has a queue that is physically associated with it means that the queue is coupled to the computer in such a way that the computer can keep utilizing the queue in case of a computer network failure, and preferably in case of any failure that is external to the computer. Queues that are physically associated with a specific computer are preferably realized by running the software, that implements the queues, on the specific computer itself, and by allocating the auxiliary storage, in which the sequence of messages or jobs of the queue is held, on the specific computer itself. In another, less preferred embodiment of the invention, the software implementing the queues does not run on the specific computer itself but on a computer (or data processing system) that is directly connected to the specific computer, i.e. not connected via the network, so that the specific computer can keep utilizing the queues in case of a network failure.

Preferably, a project management system runs on a server computer and application software runs on one or more client

computers, connected to the server computer via a computer network. The project management system on the server computer communicates with the software application packages on the client computers via the network. The software applications may relate to pre-press, to cost estimation, etc. as discussed already above. The project management system is a communication tool that allows the different players and the software applications in the Graphic Enterprise to interact formally with each other. The different players and the software applications may be located in different companies, e.g. the project management software, running on the server computer, may be in a first company while the pre-press software and the pre-press team may be in another company.

An advantage of a method in accordance with the invention is that the computers can run independently of each other, e.g. a client computer can run and execute its jobs even if the server computer is down.

Another advantage is that diverse software applications, having their own proprietary interfaces, can be controlled by the project management software.

Further advantages and embodiments of the present invention will become apparent from the following description and drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is described with reference to the following drawing without the intention to limit the invention thereto, and in which:

Fig. 1 diagrammatically shows an embodiment in accordance with the invention.

DETAILED DESCRIPTION OF THE INVENTION

Fig. 1 diagrammatically illustrates a preferred embodiment of the invention. A server computer 21 and a number of client computers 31 are interconnected by a computer network 15. Project management software runs on the server computer 21 and application

software components 37, also called client software components 37 or subsystems 37 elsewhere in this document, run on the client computers 31. The project management software automates the project and process management of the printing and publishing industry.

In Fig. 1, the computer network 15 is preferably the Internet or an Intranet, i.e. a portion of the Internet with restricted access. The server 21 and client 31 computers may be Personal Computers (PC's) or other types of data processing systems as known in the art. In the shown embodiment, the server computer 21 has three queues that are physically associated with it: a server input queue 22, a server output queue 23 and a server controlling queue 24. For each hardware node 31 connected to the network 15, or client computer 31, on which one or more application software components 37 will be installed, three local queues 32 - 34 are provided that are physically associated with their client computer 31: a client input queue 32, a client output queue 33 and a client controlling queue 34. When the project management software on the server computer 21 requires a client software component 37 to be run, or needs information from such a client software component 37, a message is put on the server output queue 23. This message is then got from the server output queue 23, transmitted, as indicated by arrow A on Fig. 1, via the network 15 to the destination client computer 31, and put on the client input queue 32 of the client computer 31. The original message may be removed from the server output queue 23. The message on the client input queue 32 is read by the client computer 31 and an action is performed on the client computer 31, e.g. a particular client software component 37 is run, or information is obtained from a particular client software component 37. Depending on the result of the action, a completion message, such as a message indicating that the particular client software component 37 ran and terminated successfully, is put on the client output queue 33. This completion message is got from the client output queue 33, transmitted, as indicated by arrow B on Fig. 1, via the network 15 to the server computer 21, and put on the server input queue 22. The completion message is read by the server computer 21 and may cause an action in the server computer 21.

As is illustrated in Fig. 1, several client software components 37 may run on the same client computer 31 (as a matter of fact, several software components may also run on the server computer 21). These client software components 37 often have their own proprietary interface and therefore need specific software to be controlled by the project management software on the server computer 21. This problem may be solved as follows. For each client software component 37 or subsystem 37 on a hardware node 31, a module that is called in this document a subsystem plug-in 36 (see Fig. 1) is run on the concerned hardware node 31. Such a subsystem plug-in 36 converts a message on the client input queue 32 to a format adapted for the subsystem 37 that is associated to the concerned subsystem plug-in 36 and for which the message is intended. Preferably, the subsystem plug-in 36 scans the input queue 32 of its hardware node 31, or client computer 31, for messages that are intended for its associated subsystem 37.

Usually, the format of the messages will depend on the type of subsystem 37. Preferably, the messages are XML-based (XML stands for eXtensible Markup Language; it is a simple and flexible text format).

In Fig. 1, only a single server computer 21 is shown. More than one server computer 21 may be used. In this document, a "topology" means a logical network of one or more server computers 21 and one or more client computers 31 deploying project management software (on the server computers 21) and client software components 37 (on the client computers 31), which software is meant to work together. Different unrelated topologies may be deployed on e.g. the same Intranet.

It is preferred that a specific process, called in this document a router process or simply a router, runs on each given hardware node 21, 31. This router process moves the messages in the output queue 23, 33 of the given hardware node 21, 31 to the destination input queue 32, 22. If the destination hardware node cannot be reached, the router process takes care of retries. For example the transmission of a message as indicated by arrow A in Fig. 1 is performed by a router process running on the server

computer 21, while the transmission of a message as indicated by arrow B in Fig. 1 is performed by a router process running on the client computer 31.

As a special case, a router may also redirect incoming messages from the local input queue to a remote input queue; in this way, one subsystem 37 may delegate a job to another subsystem 37.

Taking care that a message is transmitted to the correct destination is preferably done by the concerned router process. This may be carried out as follows. The router process inspects all outgoing messages on the local output queue, by looking at message header properties, and subsequently moves each outgoing message to the input queue related to the correct destination. This input queue is determined as follows. Preferably the software on the main server 21 and the subsystems 37 on the client servers 31 communicate with each other through each other's unique system names. The router then translates such a system name, that indicates the destination, e.g. a specific subsystem 37, to the corresponding input queue. This implies the need for some mechanism to get all routers informed at all times of the locations of all subsystems 37, and of the software running on the server computer 21. This mechanism includes a special router and the controlling queues 24, 34 that were mentioned already above in the embodiment shown in Fig. 1. The special router, called the bootstrap router, provides all the other routers with the information concerning the locations. The bootstrap router has the same functions as the other routers, which were already discussed above, and additionally the bootstrap router maintains the topology information of the system, i.e. where queues are provided, where routers are running, which (sub)systems run on what hardware nodes 21, 31, etc. When a new subsystem 37 is first started, a message is put on the controlling queue 34 of the hardware node 31 of this subsystem 37. The router of this hardware node 31 transmits this message to the controlling queue of the bootstrap router. The bootstrap router then updates its topology information and makes this information available to all routers in the system. The bootstrap router may run on a server computer 21 or on any other computer.

An advantage of this "bootstrapping mechanism" is that new subsystems 37 may be started up easily within an already established topology. It is not required to bring down the entire topology and to restart the topology after the installation of the new subsystem 37. Moreover, also a new client computer 31 with corresponding router and queues 32, 33, 34 can dynamically become part of an already established topology.

Another advantage is that subsystems 37 can easily "travel" to other client computers 31. It is not necessary to bring down the entire topology and to restart the topology after having moved the subsystem 37. If e.g. a given server 21 or client 31 computer fails, the software on that computer may have to be relocated to one or more other computers. The bootstrapping mechanism then takes care of informing all involved parties (e.g. all other subsystems 37) of the new locations.

Preferably only a single input queue 22, 32 is used per computer 21, 31 for all software applications that are part of the concerned topology. Thus, all client software components 37 on a specific client computer 31 make use of a single, shared input queue 32 (of course, 'all client software components' only means the client software components 37 that are part of the topology, and that thus communicate with a server computer 21 via messages; it does not include some arbitrary software that may also run on the specific client computer 31). More preferably, also a single shared output queue 23, 33 is used. Most preferably, each server computer 21 and each client computer 31 has a single set of queues associated with that computer and to be used for all software applications that are part of the concerned topology and that run on that computer. The set of queues includes a single shared input queue 22, 32, a single shared output queue 23, 33 and a single shared controlling queue 24, 34. An advantage of such a queue architecture is that it alleviates administration. Suppose that each subsystem 37 would read from its own queue instead of reading from a shared queue associated with the client computer 31 that the subsystem 37 is running on. In that case, adding a new subsystem 37 would require

introducing the deployment of new, previously non-existing queues into an already up-and-running configuration.

Preferably, the queues are persistent queues; this means that their contents is preserved, even in case the computer system goes down for some reason. The subsystems 37 preferably log their results locally. Results that are needed by other subsystems 37 or by project management software on the server computer 21 are preferably put on the corresponding client output queue 33. An advantage is that the subsystems 37 can run independently of each other, even if the server computer 21 is down.

Those skilled in the art will appreciate that numerous modifications and variations may be made to the embodiments disclosed above without departing from the scope of the present invention.

List of reference signs

10 : computer network system
15 : network
21 : server computer
22 : server input queue
23 : server output queue
24 : server controlling queue
31 : client computer
32 : client input queue
33 : client output queue
34 : client controlling queue
36 : subsystem plug-in
37 : client software component
A : arrow
B : arrow

■

[CLAIMS]

1. A method for sending a message from a server computer (21) to a client computer (31) via a network (15), the method comprising the steps of:
   - putting said message on a server output queue (23) that is physically associated with said server computer (21);
   - getting said message from said server output queue (23) and putting said message on a client input queue (32) that is physically associated with said client computer (31);
   - getting said message from said client input queue (32) for performing an action on said client computer (31).

2. The method according to claim 1 further comprising the steps of:
   - performing an action on said client computer (31);
   - putting a completion message on a client output queue (33) that is physically associated with said client computer (31), wherein said completion message depends on a result of said performed action.

3. The method according to claim 2 further comprising the step of:
   - getting said completion message from said client output queue (33) and putting said completion message on a server input queue (22) that is physically associated with said server computer (21).

4. The method according to claim 3 further comprising the step of:
   - removing said message from said server output queue (23).

5. The method according to claim 4 wherein said server input queue (22), said server output queue (23), said client input queue (32) and said client output queue (33) are persistent queues.

6. The method according to any one of the preceding claims wherein said client input queue (32) is a shared input queue (32) for use by all client software components (37) for running on said client computer (31) and adapted for receiving said message from said server computer (21).

7. The method according to any one of the preceding claims further comprising the steps of:
- adding a new client software component (37) for running on said client computer (31);
- putting a second message on a client controlling queue (34) that is physically associated with said client computer (31);
- getting said second message from said client controlling queue (34) and putting said message on another controlling queue (24, 34) that is physically associated with another computer (21, 31) having a bootstrap router for maintaining a topology information of a system (10) comprising said server computer (21), said client computer (31), said other computer (21, 31) and said network (15).

8. A computer network system (10) comprising means for carrying out the steps of the method according to any one of claims 1 to 7.

9. A set of computer programs comprising computer program code means adapted to perform the method according to any one of claims 1 to 7 when said set of computer programs is run on a computer network system (10).

10. A computer readable medium comprising program code adapted to carry out the method according to any one of claims 1 to 7 when run on a computer network system (10).

11. A method for sending a message by a server computer (21) to a client computer (31) via a network (15), the method comprising the steps of:

- putting said message on a server output queue (23) that is physically associated with said server computer (21);
- getting said message from said server output queue (23) and putting said message on a client input queue (32) that is physically associated with said client computer (31).

12. The method according to claim 11 further comprising the steps of:
- reading a completion message on a server input queue (22) that is physically associated with said server computer (21), wherein said completion message results from an action performed on said client computer (31) in response to said message.

13. A server computer (21) comprising means for carrying out the steps of the method according to claim 11 or claim 12.

14. A method for receiving a message from a server computer (21) by a client computer (31) via a network (15), the method comprising the steps of:
- getting said message from a client input queue (32) that is physically associated with said client computer (31);
- performing an action on said client computer (31), wherein said action depends on said message;
- putting a completion message on a client output queue (33) that is physically associated with said client computer (31), for sending said completion message to said server computer (21), wherein said completion message depends on a result of said performed action.

15. The method according to claim 14 further comprising the step of:
- getting said completion message from said client output queue (33);
- putting said completion message on a server input queue (22) that is physically associated with said server computer (21).

16. The method according to claim 14 or claim 15 further comprising the steps of:
- converting by said client computer said message got from said client input queue (32) to a format adapted for a client software component (37) for running on said client computer (31);
- running said software component (37) on said client computer (31);
- determining said completion message depending on a result of said run of said client software component (37).

17. A client computer (31) comprising means for carrying out the steps of the method according to any one of claims 14 to 16.

18. A computer program comprising computer program code means adapted to perform the method according to any one of claims 11 to 12 or claims 14 to 16 when said program is run on a computer (21, 31).

19. A computer readable medium comprising program code adapted to carry out the method according to any one of claims 11 to 12 or claims 14 to 16 when run on a computer (21, 31). ■

[ABSTRACT]

METHOD FOR SENDING MESSAGES IN A COMPUTER NETWORK

A method and a system (10) for sending a message from a server computer (21) to a client computer (31) via a network (15), the method including (a) putting the message on a server output queue (23) that is physically associated with the server computer (21); (b) getting the message from the server output queue (23) and putting the message on a client input queue (32) that is physically associated with the client computer (31) and (c) getting the message from the client input queue (32) for performing an action on the client computer (31).

Fig. 1

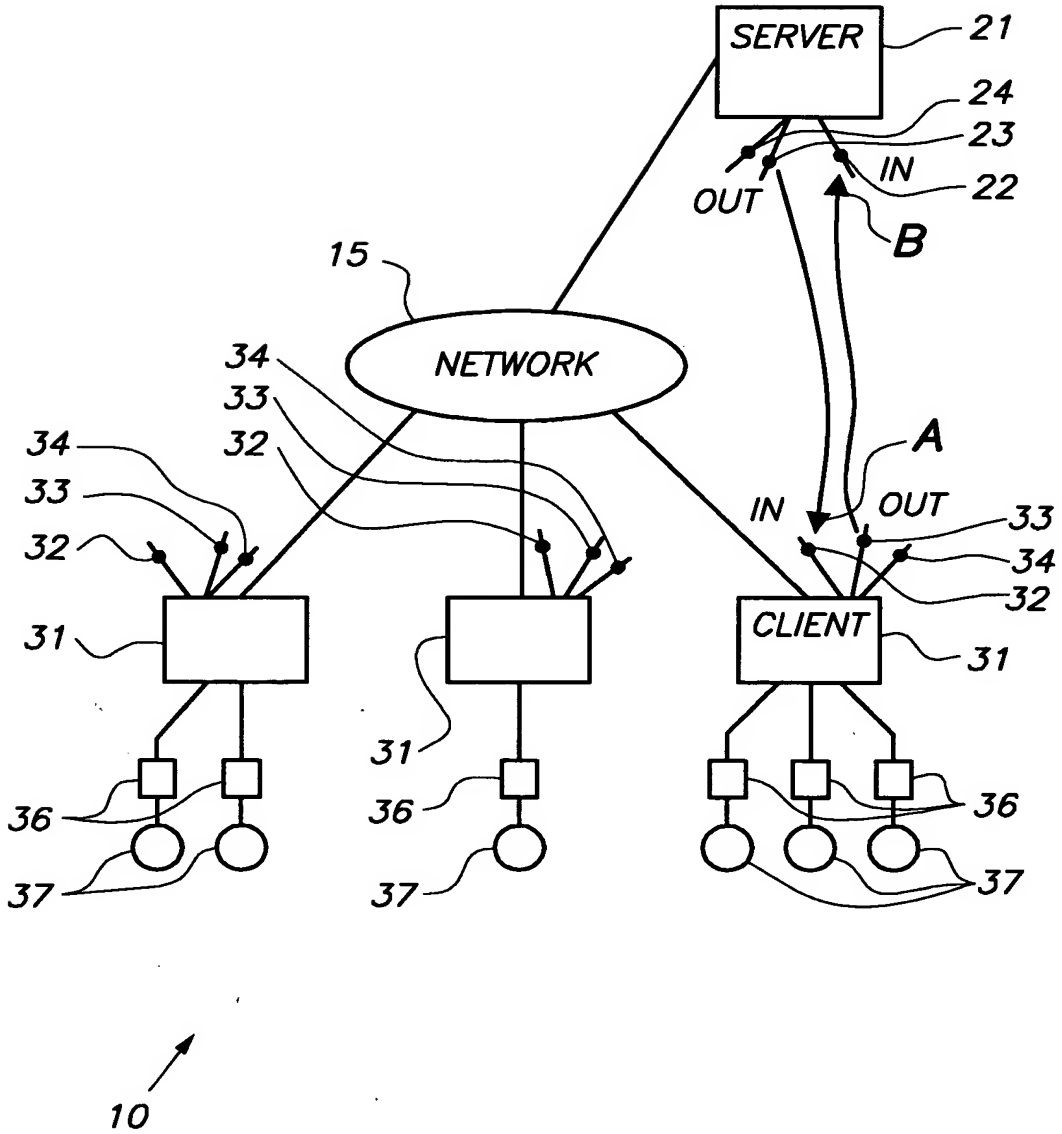
SERVER
21
24
23
IN
22
OUT
B
15
NETWORK
34
33
32
A
IN
OUT
33
34
32
CLIENT
31
31
31
36
36
36
37
37
37
10

FIG. 1